Interfacing with network elements for network management and control

Chaganty, S. Reeves, D.A. Green, E.
GTE Telecom Inc., Bothell, WA;

This paper appears in: **Communications, Computers and Signal Processing, 1991., IEEE Pacific Rim Conference on**
Meeting Date: 05/09/1991 -05/10/1991
Publication Date: 9-10 May 1991
Location: Victoria, BC , Canada
On page(s): 685-687 vol.2
References Cited: 2
IEEE Catalog Number: 91CH2954-6
INSPEC Accession Number: 4208868

Abstract:
One approach to solving the variety of problems involved with interfacing to an assortment of network elements (NEs) is to provide a network interface system that uses an event control philosophy. GTE Telecom has implemented this philosophy by developing a system known as the network interface system (NIS). The NIS enables an end user to logically connect through a terminal to any NE in the network that supports communications. The system uses a GTE Telecom-developed programming language to manage and control events. This language is called the network interface module language (NIML). This generic interface responds to event status changes and supports the data collection activities required by the host application managing the network. The architecture of the NIS is modular to maximize the amount of generic code. NIS can adapt itself to the unique environment required for a specific NE by loading a NIML object at run time that will manage the events of that particular interface

Index Terms:
GTE Telecom NIML computer interfaces data collection activities event control philosophy event status changes generic code generic interface high level languages host application modular architecture network control network elements interface network interface module language network interface system programming language telecommunication network management telecommunications computer control telecommunications computing GTE Telecom NIML computer interfaces data collection activities event control philosophy event status changes generic code generic interface high level languages host application modular architecture network control network elements interface network interface module language network interface system programming language telecommunication network management telecommunications computer control telecommunications computing

IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, May 9-10, 1991

INTERFACING WITH NETWORK ELEMENTS FOR NETWORK MANAGEMENT AND CONTROL

Srinivas Chaganty, Donald A. Reeves and Evan Green
GTE Telecom Incorporated.
22027 17th Avenue SE
Bothell, Washington, USA

Abstract

Managing today's networks is necessary to optimize the use of the network resources and to preserve the traffic handling and routing capabilities of the network when it is being stressed beyond its engineering capacity. Networks can be viewed as a collection of Network Elements (NE). NE's are devices such as hosts, gateways, modems, muxes, PBXs, terminal servers, etc. In order for the network manager to control the network, traffic information (i.e., surveillance counts, state indications, reference data) about a particular NE, or group of NE's, is essential. There are a variety of NE's on the market today, designed and manufactured by a variety of vendors. The problem for today's NE integrators and traffic information collectors is to provide a method for interpreting the NE "language" into information recognizable to the traffic collection system. The Network Interface System (NIS) developed by GTE Telecom Incorporated demonstrates one method of solving this problem. This paper will discuss the interface problem, analysis techniques and solution development.

Introduction

Interfacing with the elements of a communication network is essential for network administrators to efficiently manage their network resources. Management of the network facilities is dependent on the collection of operational information (surveillance counts, state indications, reference data) normally referred to as traffic data, about each element of the network. A typical network consists of multi-vendor network elements each supporting a different kind of interface. The current dilemma for builders of data collection systems is how to provide network interface services for the wide variety of network elements that may exist in a customer's network. Evolving international standards, along with market demands, will eventually produce NE's that can inter-operate via an "universally accepted language." Until that time, providers of network interface services must be able to accommodate the multiplicity of interface requirements either by developing unique solutions for each interface or designing generic solutions that address various network element types.

A Solution

One approach to solving the variety of problems involved with interfacing to an assortment of network elements is to provide a network interface system that uses an event control philosophy. GTE Telecom has implemented this philosophy by developing a system known as the Network Interface System (NIS). The NIS enables an end user to logically connect through a terminal to any NE in the network that supports communications. The system uses a GTE Telecom-developed programming language to manage and control events. This language called Network Interface Module Language (NIML), is used to write a programmatic solution that allows the interface to provide communication services between the host application and the NE's. This generic interface responds to event status changes and supports the data collection activities required by the host application managing the network.

The architecture of the Network Interface System is purposefully modular to maximize the amount of generic code. NIS can adapt itself to the unique environment required for a specific NE by loading a NIML object at run time that will manage the events of that particular interface.

Network Interface System (NIS)

The NIS has two distinct but related domains (environments). The first is the NIML development domain or the NIML Stream Analyzer (SA). The SA is a collection of functions and software tools used to analyze communication data streams, generate and test NIML programs and manage releases of NIML objects. The Network Interface Control (NIC) domain is the production environment. All of the components of the NIC work together to provide the host application with the NE interface services.

NIML Stream Analyzer

When developing NE interfaces, the developer must undertake several tasks to provide a fully-functional interface. Basically, creating an interface requires

CH2954-6/91-0000-685 $1.00 © 1991 IEEE

analysis of the data exchange over the interface, study of the nature of the data to understand the objectives of the communication, application of this understanding to develop an automated interpreter for the data (i.e., NIML program), and application of the solution developed to sample data streams to verify that it performs as expected. The Stream Analyzer tool set was developed to provide the interface developer with all of the capabilities required.

The Network Interface Module Language (NIML) was developed to allow event control logic to be imbedded into the generic NIC. The NIML language is based on two concepts. First, it is equipped with a primitive loop, much like the RPG family of languages. This loop provides a logical connection with an "engine" that supports execution and provides services such as I/O and inter process communications. Second, NIML is made up of a simple set of instructions and a large set of powerful functions as is the "C" language. This facilitates the simple logic construction of NIML programs. The library of NIML functions is oriented strongly to the activities associated with processing data streams received from and sent to network elements.

NIML logic revolves around a primary event control loop. This loop is the primary interface between the NIML program and the facilities of the engine that communicates with the world beyond. As such, the event control loop is the fundamental control mechanism for execution. The primary purpose of the NIML language and engine is to manage events.

Within NIML there are many events that are defined and can occur. Since events are a primary element of NIML logic, they are divided into three classifications to help distinguish their properties. The first group of events is called "timeouts." The second group is called "predefined" events. Each of the predefined events is associated with a specific construction in the NIML engine. Examples of these types of events include communication errors, data interrupts, and initial session requests. The third group of events is referred to as "user-defined" events. The NIML programmers can assign meanings to them within the NIML program and use them to control logic flow within the program itself. To allow control over how events may be related to each other, the NIML programmer has the flexibility of establishing event priorities.

During execution the event controller examines the event table looking for events that have been "turned on." When such an event is found, the event controller initiates processing the NIML program at the location specified by the label associated with the event. The event controller services each event based on the priority and logic supplied by the programmer.

Network Interface Control (NIC)

The NIC is composed of five modules that have specific functions. Together with the NIML object they form the basic production domain or system. Figure 1 indicates the relationship between the various components of the NIC.

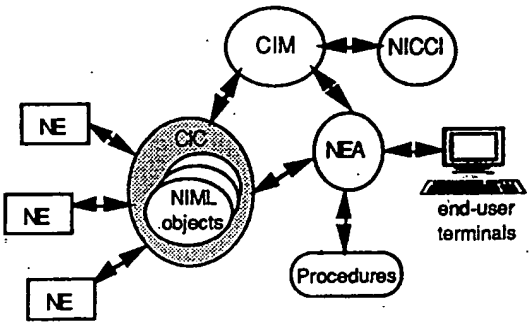

Figure 1

The Network Element Access (NEA) module displays prompts to the end-user, accepts responses (commands) from the end-user and displays framed messages received from the NE. Data is presented in three different modes: "records" are framed messages labeled with a message ID and sent on to a message collection and distribution application; "display" is data shown on an end-user terminal for display purposes only; and "prompt" is presented to only one end-user conversing with the NE. While only one end-user receives prompts, there may be numerous end-users presented with displays and records.

The Communications Interface Manager (CIM) manages all of the Network Interface resources, maintains a table of the current resource status. It is also responsible for granting (or queuing) all session requests, assigning identification numbers to each session, and handling all terminate session requests.

The CIC component of the NIC performs a variety of tasks. The CIC Application Layer first receives control from the CIM and passes information regarding the session path and type of NE to the CIC Engine Layer. The CIC Engine Layer then loads the appropriate NIML object file for the specified NE. The NIML object file translates the NE specific communication protocols into NI operation codes, thereby isolating them from the rest of the system and supporting the general design

goal of maximizing the amount of generic code while optimizing the portability of the NIML objects.

Message Handling

Once the session is established, data received from the NE is passed from the CIC Physical Layer to the CIC Engine. The CIC Engine examines the data received, frames the message according to the protocol rules defined in the NIML object and deposits the data in a message log file. When done, the CIC Engine informs the CIC Application Layer of the existence and location of the message. The CIC Application Layer then informs the NEA and the message distribution module of the existence of the message so that NEA can display it to the end-user and the message can be distributed to other applications.

When data is to be sent from the end-user to the NE, the NEA deposits the data into the data file and notifies the CIC Application Layer of the existence and location of the data. The CIC Application Layer then instructs the CIC Engine to read the data from the data file and pass it along to the NE. The CIC Engine first examines the data, formats it for the NE based on the protocol defined by the NIML object and passes it to the Physical Layer for transfer to the NE.

The NIS operator uses the Network Interface Command Control Interpreter (NICCI) to communicate with the CIM and/or CICs. The operator can display and/or change the dynamic configuration of the Network Interface System. Specifically, the operator can start or stop CIC activity, exchange commands and responses with CIM or directly communicate with the CICs. In addition, the operator can request statistical NIS measurement information from measurement points being maintained within the NIS.

Conclusions

Currently there is a very large installed base of NE's that does not provide a standard interface, especially in telephony environment. Although the existing nonstandard devices will be phased out very gradually over a long period, customers will require a management system today that manages these nonstandard devices. Therefore, the management systems are required to interface with a variety of multi-vendor network elements. This mandates a well architected interface system that allows a management system to remotely access various NE's for controlling them and gathering data from them. GTE Telecom's NIS furnishes a complete solution to the network element interface question. Stream Analyzer of the NIS provides tools for "reverse-engineering" interface protocols and developing software solutions. To support the production environment, the Network Interface Control Modules only require loadable objects to adjust their logic. All in all, the Network Interface System provides an efficient and productive environment for creating network interface solutions.

Acknowledgment

The Network Interface System was conceived and developed by several individuals for GTE Telecom Incorporated, Network Management & Control (NM&C), Bothell, Washington. The authors wish to acknowledge the contribution of those individuals.

References

[1] _____, Network Traffic Management (NTM) Operations System (OS) Requirements, Technical Advisory TA-TSY-000753, Issue 2, Bell Communications Research, Inc, December 1988.

[2] Donald A. Reeves, "Traffic Issues Involved in a Telecom Environment," presented at the Northcon/90, Seattle, Washington, October 9 -11, 1990.